IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of	:	
Kimito HORIE	:	
Serial No. NEW	:	**Attn: APPLICATION BRANCH**
Filed February 24, 2004	:	Attorney Docket No. 2004-0214A

DATA PROCESSING APPARATUS AND
DATA DECODING APPARATUS

CLAIM OF PRIORITY UNDER 35 USC 119

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Applicant in the above-entitled application hereby claims the date of priority under the International Convention of Japanese Patent Application No. 2003-045601, filed February 24, 2003, as acknowledged in the Declaration of this application.

A certified copy of said Japanese Patent Application is submitted herewith.

Respectfully submitted,

Kimito HORIE

By _____
Nils E. Pedersen
Registration No. 33,145
Attorney for Applicant

NEP/krg
Washington, D.C. 20006-1021
Telephone (202) 721-8200
Facsimile (202) 721-8250
February 24, 2004

THE COMMISSIONER IS AUTHORIZED TO CHARGE ANY DEFICIENCY IN THE FEES FOR THIS PAPER TO DEPOSIT ACCOUNT NO. 23-0975

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００３年　２月２４日

出願番号
Application Number: 特願２００３－０４５６０１

[ST.10/C]: [ＪＰ２００３－０４５６０１]

出願人
Applicant(s): 沖電気工業株式会社

２００３年　６月　２日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証番号　出証特２００３－３０４２８０７

【書類名】　特許願
【整理番号】　SA003792
【あて先】　特許庁長官殿
【国際特許分類】　G06F　5/00
　　H03M　7/30
【発明者】
　【住所又は居所】　東京都港区虎ノ門１丁目７番１２号　沖電気工業株式会社内
　【氏名】　堀江　公人
【特許出願人】
　【識別番号】　000000295
　【氏名又は名称】　沖電気工業株式会社
【代理人】
　【識別番号】　100082050
　【弁理士】
　【氏名又は名称】　佐藤　幸男
【手数料の表示】
　【予納台帳番号】　058104
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
　【包括委任状番号】　9100477
【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　データ圧縮装置及びデータ展開装置

【特許請求の範囲】

【請求項１】　データ列から生データとポインタと命令とを出力するデータ圧縮装置であって、

前記命令が対象とするデータ列が、前記生データの対象とするデータ列と前記ポインタが指示するデータ列、または、これらの組合せと一致する場合に、前記生データまたはポインタに置き換えて命令を出力する置き換え手段を備えたことを特徴とするデータ圧縮装置。

【請求項２】　請求項１に記載のデータ圧縮装置において、

命令に割り当てられたコード長が、生データとポインタとからなるデータ長よりも少ない場合、その元となる生データまたはポインタを命令に置き換える置き換え手段を備えたことを特徴とするデータ圧縮装置。

【請求項３】　請求項１または２に記載のデータ圧縮装置において、

命令は、命令コードと拡張コードで構成されることを特徴とするデータ圧縮装置。

【請求項４】　請求項４に記載のデータ圧縮装置において、

拡張コードは、命令の種類を表すコードとオペランドとで形成されることを特徴とするデータ圧縮装置。

【請求項５】　請求項１～４のいずれかに記載のデータ圧縮装置において、

第１のビットを生データとポインタと命令とを分別することに用い、

第２のビットをポインタと命令とを分別することに用いることを特徴とするデータ圧縮装置。

【請求項６】　請求項５に記載のデータ圧縮装置において、

複数の命令が、それぞれ命令の種類とオペランドにより構成される場合に、各命令の種類に応じて漸次コード長を増加させるコード化を行う置き換え手段を備えたことを特徴とするデータ圧縮装置。

【請求項７】　請求項５に記載のデータ圧縮装置において、

複数の命令が、それぞれ命令の種類とオペランドにより構成される場合に、各

オペランドのパラメータに応じて漸次コード長を増加させるコード化を行う置き換え手段を備えたことを特徴とするデータ圧縮装置。

【請求項８】　請求項１～７のいずれかに記載のデータ圧縮装置において、

特定のポインタが指定するデータ列と、他の複数のポインタが指定するデータ列とが一致する場合に、前記特定のポインタを定義命令に置き換え、かつ、前記他の複数のポインタを前記定義命令に対応するコード置き換え命令に置き換える処理を、所定のデータ単位毎に行う置き換え手段を備えたことを特徴とするデータ圧縮装置。

【請求項９】　請求項８に記載のデータ圧縮装置において、

特定のポインタが指定するデータ列として、所定のデータ単位内に存在する当該データ列のうち、最初に出現したものを選択することを特徴とするデータ圧縮装置。

【請求項１０】　請求項８または９に記載のデータ圧縮装置において、

複数の定義命令を設定した場合に、当該複数の定義命令に置き換えたポインタが指示していた複数のデータ列の、所定のデータ単位内での出現頻度を計数し、当該出現頻度の順番に、前記定義命令を記載した定義表を作成する置き換え手段を備えたことを特徴とするデータ圧縮装置。

【請求項１１】　請求項１０に記載のデータ圧縮装置において、

定義表は所定のデータ単位毎に新たに作成することを特徴とするデータ圧縮装置。

【請求項１２】　請求項８～１０のいずれかに記載のデータ圧縮装置において、

特定のポインタが指定するデータ列のデータ量と、他のポインタが指定するデータ列のデータ量とが一致するかを判定し、かつ、所定のデータ単位における前記特定のポインタの指定するデータ列のアドレスと、前記他のポインタの指定するデータ列のアドレスとの差が、前記特定のポインタの持つオフセット値と、前記他のポインタの持つオフセット値との差に一致するかを判定し、一致した場合に、前記特定のポインタが指定するデータ列と、前記他のポインタが指定するデータ列とが一致すると判定する置き換え手段を備えたことを特徴とするデータ圧

縮装置。

【請求項１３】　生データとポインタと命令とを含むデータが入力された場合、

前記命令を実行して、生データまたはポインタに戻すと共に、前記生データまたはポインタを対象とするデータ列に戻すよう構成されたことを特徴とするデータ展開装置。

【請求項１４】　生データとポインタと命令とを含むデータで、かつ、特定のポインタが指定するデータ列と、他の複数のポインタが指定するデータ列とが一致する場合に、前記特定のポインタを定義命令に置き換え、かつ、前記他の複数のポインタを前記定義命令に対応するコード置き換え命令に置き換えるよう所定のデータ単位毎に圧縮処理されたデータが与えられた場合、

前記所定のデータ単位毎に、コード置き換え命令を定義命令に戻し、かつ、定義命令を生データまたはポインタに戻し、これら生データまたはポインタを対象とするデータ列に戻すよう構成されたことを特徴とするデータ展開装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、ＬＺＳＳ等のデータ圧縮・展開方法を拡張したデータ圧縮装置及びデータ展開装置に関する。

【０００２】

【従来の技術】

近年のインターネットの普及に伴い、データをネットワークを介して短時間に伝送する技術が進展しており、データ圧縮方式は、伝送するデータそのものを少なくすることでこの目的を達成しようとする。

【０００３】

このような圧縮方式において、ロスの出ない（lossless）データ圧縮方式（compression method）の中で、現在の最も多く利用されている辞書型（dictionary coding）の起源は、Abraham Lempel氏とJacob Ziv氏が１９７７年に創作したLempel-Ziv符号化法であり、これはスライド辞書法またはＬＺ７７などと呼ばれて

いる。このＬＺ７７は、以前に符号化した入力文字データ列を辞書として利用し、入力文字データ列中の連続した部分との最長一致を検索し、その一致情報を符号化するというものである。ＬＺ７７は、ヌルポインタ（the null-pointer）を有し、かつ、マッチ（match）後に出力する最初の文字（character）が次のマッチにも使えるので、その場合には出力をする必要がないということから多少の冗長度があり、その後、これを改良する方式として１９８２年にStorer氏とSzymanski氏がＬＺＳＳというデータ圧縮方式を創作した。そして、このようなＬＺＳＳを用いた各種の技術が提示されている（例えば、特許文献１、２、３参照）。

【０００４】

ＬＺＳＳは、その文字列が一定のポインタ長よりも長いときにのみそのポインタを出力し、それ以外の時には文字データを出力するという方式である。そのため、ＬＺＳＳのコード化では、コード化した文字データとポインタとが混じった形の出力データとなるため、出力中の文字データとポインタとを区別するために特定のビット（an extra ID-bit）を用いている。

【０００５】

ＬＺＳＳの圧縮率を他の圧縮方式と比べた場合、特に規則性のある文字データに対する圧縮率が高いことが知られており、また、その展開（decoding）が高速であるという特徴がある。例えば、上記のポインタを他の統計型圧縮方式（the entropy methods）と組み合わせた方式でコード化する場合に特にその効果が大きい。

【０００６】

【特許文献１】

特開平５－２４１７７７号公報

【特許文献２】

特開平７－２７３６６７号公報

【特許文献３】

特開２０００－３１５９５４号公報

【０００７】

【発明が解決しようとする課題】

しかしながら、上記従来のデータ圧縮方法は、文字データとポインタで構成されているため、文字データ以外のデータ、例えば、パスワードや著作権情報といった情報を埋め込むのは困難であった。また、文字データとポインタのみで構成されているため、これらの手段の範囲内で更に優れた圧縮率を達成するには、限界があった。

【０００８】

【課題を解決するための手段】

本発明は、前述の課題を解決し、自由な圧縮が行えるデータ圧縮装置を実現するため、データ列から生データとポインタと命令とを出力するようにしたものである。即ち、命令が対象とするデータ列が、生データの対象とするデータ列とポインタが指示するデータ列、または、これらの組合せと一致する場合に、生データまたはポインタに置き換えて命令を出力するようにしたものである。

【０００９】

【発明の実施の形態】

以下、本発明の実施の形態を具体例を用いて詳細に説明する。

先ず、具体例の説明に先立ち、本発明の原理について説明する。

【００１０】

本発明は、上記の課題を解決するため、出力データ列中に文字データおよびポインタ（pointer）の他に新たに命令（command）を追加し、文字データ若しくはポインタ、更には命令自体をも制御することにより、全体として新たな機能追加と圧縮率の向上を実現するデータ圧縮装置を提供するものである。

【００１１】

本発明のデータ圧縮装置は、上述したように、圧縮後のデータに文字データおよびポインタの他に新たに命令を追加したことに特徴がある。この命令は様々に定義できるが、基本的には上記の文字データを出力するために使われる。例えば、ポインタ反復命令は、該当するポインタを複数回反復し、その結果、そのポインタの出力である文字データが複数回反復して出力される。この反復回数が少なく、かつ、一回に出力される文字データの個数が少ない場合には、上記命令を追加したために却ってデータ量が増え、その結果、圧縮率が低下することがある。

本発明ではこのような場合は命令の追加を行わない選択をする（後述する具体例１のステップＳ４の動作参照）。従って、原則として、本発明では従来のＬＺＳＳ等の圧縮率よりもその圧縮率を低下させることはない。もっとも、例外的に、著作権情報等の埋め込みのためにこの圧縮率の低下を許容することがある。また、命令を追加し、若しくは追加しなかった場合でも、展開（decoding）されたデータ出力は何ら変更がないことに注意したい。埋め込まれた著作権情報等は直接的に何らデータ出力を変更しない。これは従来の圧縮方式を拡張する場合に常に留意すべき点である。

【００１２】

データ圧縮における上記原理は、データ圧縮が従来のように順序のついた（sequential）文字列を順序のついた文字データとポインタに変更するという直接的な視点から一歩立ち上がって、データ圧縮が順序のついた文字列を「プログラム」に変更するという新たな視点に立とうとするものである。この視点に立つ第一歩は、ポインタというものが一種の命令であると考えると理解しやすい。ＬＺＳＳ等の圧縮方式におけるポインタ（Ｂ，Ｌ）の意味は、後述するように、Ｂ文字分文字列を遡ってＬ文字分出力しなさい、という意味である。これは正に一種の命令である。半導体の分野では論理回路をプログラムで記述する方法が普及しており、Velilog等の言語が知られている。これは回路構造をプログラムとして出力するものであるが、本発明ではデータ構造をプログラムとして出力するものである。

【００１３】

図１は、本発明のデータ圧縮装置の実施に当たって、追加エンコーダ（図中の置き換え手段２に相当する）を導入した構成を示すものである。この構成の詳細については後述するが、図示のデータ圧縮装置は、入力バッファにある入力データ１１（文字列）を、従来の圧縮装置に相当する圧縮装置１により順序のついた文字データおよびポインタに変換して中間バッファに出力した後、本発明で導入した追加エンコーダ（置き換え手段２）により更に順序のついた文字データ、ポインタおよび命令を含む出力データ１３に変換して出力バッファに出力する。ここでは、従来の圧縮装置１を本発明で導入した追加エンコーダに含めて考えるこ

とも可能である。そして、順序のついた文字データ、ポインタおよび命令の出力とは、正にプログラムの生成そのものである。

【００１４】

次に、文字列から文字データとポインタとを出力するデータ圧縮方式の一例としてＬＺＳＳのコード化について説明する。

【００１５】

図２は、ＬＺＳＳのコード化（encoding）の説明図である。

図２では、文字列「ＡＡＢＢＣＢＢＡＡＢＣ」をポインタ長２バイトでコード化する例を示している。このような場合、ＬＺＳＳでは次のように行う。

(1)文字列の最初の文字「Ａ」は、最初の文字であって圧縮の対象にならないので、その生データ「Ａ」を文字データとして出力する。

(2)文字列の２番目の文字「Ａ」は、既に出現している(1)の最初の文字「Ａ」と一致ずるが、ポインタ長が２バイトであり、２バイト分の一致を見ていないのでこの場合もその生データ「Ａ」を文字データとしてそのまま出力する。

【００１６】

(3)文字列の３番目の文字「Ｂ」は、新しく出現した文字であり、(1)と同様に扱う。

(4)文字列の４番目の文字「Ｂ」は、直前に出現した文字であり、(2)と同様に扱う。

(5)文字列の５番目の文字「Ｃ」は、新しく出現した文字であり、(1)と同様に扱う。

(6)文字列の６番目の文字「Ｂ」は、前に出現した文字であるが、その一つ後の７番目の文字「Ｂ」と合わせて「ＢＢ」と一致し、この場合にはポインタ長の２バイト以上で一致し、また、文字列の８番目の文字「Ａ」と合わせた文字列「ＢＢＡ」はまだ出現していないので、ポインタ（３，２）を出力する。ここで、ポインタ（Ｂ，Ｌ）の意味は、Ｂ文字分文字列を遡ってＬ文字分出力しなさい、という意味である。

【００１７】

(7)文字列の９番目から始まる文字列「ＡＡＢ」は、第１番目から始まる三つ

の文字列に一致し、その長さは２バイトのポインタ長以上であるので、ポインタ（７，３）が出力される。

(8)文字列の１１番目の文字「C」は、前に出現した文字であるが、ポインタ長よりも短いため、(2)と同様に扱う。

【００１８】

図３は、ＬＺＳＳ等で利用されるコード化の説明図である。

このコード化は、出現頻度の高い短いデータ長のポインタに短いコード長のコードを割り当てるものである。

図３において、出力データ（Compressed Stream）１０１は、圧縮されたデータ列（Compressed String）１０２と終端マーカ（End Marker）１０３から構成される。圧縮されたデータ列１０２は、先頭のビットが０＋生データ（Raw Byte）１０４か、若しくは、先頭のビットが１＋圧縮コード（Compressed Byte）１０５の集合である。尚、この「先頭のビット」とは、上述した特定のビット（an extra ID-bit）のことである。

【００１９】

ここで、生データ１０４は、ＡＳＣＩＩのような８ビット（１バイト）で構成されるが、圧縮コード１０５はオフセット（Offset）１０６とコード長（Length）１０７で構成される。即ち、このオフセットとコード長が上述したポインタ（Ｂ，Ｌ）のＢとＬに相当する。オフセット１０６は、コード化の効率を考慮して、先頭ビットが１である７ビットオフセット（合計８ビット）、若しくは、先頭ビットが０である１１ビットオフセット（合計１２ビット）を用意している。即ち、オフセットの値が小さい場合は７ビットオフセットを用い、この７ビットオフセットでは表せない場合に１１ビットオフセットを用いる。尚、この１１ビットオフセットは、入力バッファの容量である２Ｋバイトに基づくものである。

【００２０】

コード長１０７は、出現頻度の高い短いデータ長のポインタに短いコード長のコードを割り当てる上述した統計型圧縮方式である。また、終端マーカ１０３は、オフセットが０の９ビットコード０ｘ１８０に割り当てている。尚、０ｘ１８０は、１６進数表記の１８０を表しており、２進数表記の場合は、図示のように

、１１００００００００となる。

【００２１】

尚、この発明では主として文字列の圧縮の対象とするが、上記のコード構成からも入力データは一つの単位が８ビットであれば文字列に限定されないことは明らかである。

【００２２】

本発明の課題は、もっと自由な圧縮、更に優れた圧縮率を持つ方式を可能にするようＬＺＳＳ等の圧縮方式を改良することであり、その新たな方法が従来の圧縮方法の利点を損なわないようにすることが重要である。そのため、ＬＺＳＳ等の拡張に際し、特に命令の追加に際し、本発明では、その命令のコード化について従来の方法を拡張するようにしている。具体的には、図３で示したコード化の方法を拡張する。

【００２３】

統計型圧縮方式を加味したコード化は、一見拡張の余地がないようであるが、唯一特殊な命令、終端マーカ（図３の１０３）が使われている。従来のコード化では終端マーカは、オフセットが０の９ビットコード０ｘ１８０に割り当てているが、本発明の具体例１ではこれを拡張し、命令として用いる。

【００２４】

《具体例１》

〈構成〉

図１は、本発明のデータ圧縮装置の具体例１の構成図である。

図示の装置は、圧縮装置１、置き換え手段２からなる。圧縮装置１は、例えば従来のＬＺＳＳ等の圧縮装置であり、入力データ１１から、中間データ（文字データとポインタ）１２を出力する。置き換え手段２は、本具体例における追加エンコーダであり、圧縮装置１から出力された中間データに対して命令を追加し、出力データ（文字データ、ポインタ、命令）１３を出力する機能を有している。

【００２５】

図４は、具体例１における命令の基本構成を示す説明図である。

本具体例では、図４（ａ）に示すように、全ての命令は終端マーカ２１と拡張

コード２２から構成される。終端マーカ２１は、従来の終端マーカ（図３の終端マーカ１０３に相当するコード）と、同様であって、オフセットが０の９ビットコード０ｘ１８０をそのまま使うことにする。尚、このコードは図３に示した一般的なコード方式にならって採用したに過ぎず、これに限定されるものではない。ただ、どのようなコードを選択するにせよ、コード化の際に文字データおよびポインタに割り当てられたコードと抵触しないものを選択する必要がある。

【００２６】

従来のデータ展開方式では、圧縮された文字データ等が展開され、かつ、終端マーカを検出したとき、そのデコーダ（decoder）は展開を停止する。しかしながら、本具体例のデコーダは、その終端マーカを拡張命令と解釈し、その命令を実行する。本具体例では、図４（ｂ）に示すように、拡張コード２２を、命令の種類２２ａとオペランド２２ｂで構成する。ここで、命令の種類２２ａは、当然、従来の終端マーカ（図３中の終端マーカ１０３に相当する）を含むものでなければならない。多くの命令を必要とする場合は、この拡張コード２２のビット数を増やして対応する。オペランド２２ｂは、その命令の種類２２ａに応じたパラメータを指定する部分で、命令に対する引数として表示してある。

【００２７】

図５は、命令の作成例を示す説明図である。

本具体例では、命令の種類に４ビット割り当て、１６種類の命令を作れるよう構成した。

先ず、命令の種類が０ｘ０はＥＮＤ命令であり、従来の終端マーカと同様、出力データの終了を意味する。この命令は全体で１３ビットとなる。

【００２８】

命令の種類が０ｘ１は、ＲＤ（Repeat Data）命令であり、データ制御命令である。ＲＤ（Ｂ，Ｌ，Ｎ）は、Ｂ個前のＬ個の文字列をＮ回繰り返すという内容を持つ。パラメータＢには７ビット、パラメータＬには８ビット、また、パラメータＮには４ビットを割り当てているので、全体では３２ビットの命令になる。「Ｂ個前」といったとき、ここではその中には文字データだけでなく、ポインタや命令も数えるものとする。もちろん、文字だけを計数する命令を作成すること

もできる。Ｂ個前のＬ個のデータの中に文字以外のポインタや命令が含まれるとき、この命令に関する限り、デコーダエラーとなる。ＲＤ（Repeat Data）命令は、従来のポインタを一種の命令と考えたときに、その命令を拡張した機能を持つことになる。

【００２９】

命令の種類が０ｘ２は、ＲＰ（Repeat Pointers）命令であり、ポインタ制御命令である。ＲＰ（Ｂ，Ｎ）は、Ｂ個前のポインタをＮ回繰り返すという内容を持つ。パラメータＢには７ビット、パラメータＮには４ビットを割り当てているので、全体では２４ビットの命令となる。「Ｎ回繰り返す」とは、そのポインタが出力する文字データをＮ回繰り返して出力するという意味である。この使い方は、ポインタ自体を単語と見なし、その単語を繰り返して出力する、というのに似ている。この見地によれば、異なる出力を与えるポインタ群は単語を集めた辞書に相当するから、辞書には一つの単語が記載されていれば十分である。Ｂ個前のデータがポインタ以外のとき、この命令に関する限りデコーダエラーとなる。

【００３０】

命令の種類が０ｘ３は、ＲＤＰ（Repeat Data and Pointers）命令であり、文字データおよびポインタの制御命令である。ＰＤＰ（Ｂ，Ｌ，Ｎ）は、Ｂ個前のＬ個のデータ列をＮ回繰り返すという内容を持つ。パラメータＢには７ビット、パラメータＬには８ビット、及びパラメータＮには４ビットを割り当てているので、全体では３２ビットの命令となる。同様に、「Ｂ個前」といったとき、その中には文字データだけでなく、ポインタや命令も数えるものとする。この場合も、「Ｂ個前」といったとき、文字だけを計数する命令を作成することもできる。数えられた命令の中に文字データを出力する命令等が存在したとき、本具体例のデコーダは、その出力をも展開しなければならない。コード化は容易であるが、デコードの際に処理負担が増大する。

【００３１】

命令の種類が０ｘ４はＯＭＤ（Output Modified Data）命令であり、データ制御命令である。ＯＭＤ（Ｂ，Ｌ，Ｍ，Ｃ）は、Ｂ個前のＬ個のデータ列のうちＭ番目の文字データをＣに変更して出力するという内容を持つ。パラメータＢには

７ビット、パラメータＬには８ビットを割り当て、更に、パラメータＭには７ビット、パラメータＣには８ビットを割り当てているので、全体では４３ビットの命令になる。この命令の存在意義は、同一類似の文字列が入力した際に、１文字違いであれば、この命令で訂正した方が簡単である、というところにある。この命令を使うことによりデータ量が増加する場合、本具体例では採用しないよう構成されている。

【００３２】

命令の種類が０ｘ５は、ＣＰ（Connect Pointers）命令であり、ポインタ制御命令である。ＣＰ（Ｂ１，Ｂ２）は、Ｂ１個前のポインタが指す文字列と、Ｂ２個前のポインタが指す文字列とを結合して出力するという内容を持つ。パラメータＢ１およびＢ２にはそれぞれ７ビットを割り当てているので、全体では２７ビットの命令になる。この命令の存在意義は、二つのポインタを続けて出力するよりは全体のビット数が減少するというところにある。ただ、削減された後のポインタを他のポインタ制御命令で直接することはできなくなる、という問題が発生する。この場合に、そのポインタの代わりに、その分エンコーダに負担がかかるが、命令自体を解読処理する命令を使えば間接的な利用が可能である。しかし、そのポインタが同一の文字列を指す二番目以降のポインタであるならばこのような問題は発生しない。この命令を使うことによりデータ量が増加する場合、本具体例では採用しない。また、指し示すデータがポインタ以外の時にはデコーダエラーになる。

【００３３】

命令の種類が０ｘｄは、ＳＰＷ（Set Password）命令であり、パスワード設定命令である。ＳＰＷ命令は、これに続くオペランドにパスワードを埋め込む。本具体例では、パスワードに８バイトを割り当てているので、全体で７７ビットの命令になる。パスワードの検証は図示しないアプリケーションプログラムにより最初に行われ、認証が得られない場合にはデコーダによる展開は禁止される。この命令の挿入によりデータ量が増加するときでも、この命令を削除する権限を、置き換え手段２に持たせないよう構成することが必要である。

【００３４】

命令の種類が０ｘｅは、ＳＣＲ（Set Copyright）命令であり、著作権情報設定命令である。ＳＣＲ命令は、これに続くオペランドに著作権情報を埋め込む。本具体例では、著作権情報に８バイトを割り当てているので、全体では７７ビットの命令になる。著作権情報は、その著作権の権利者さえ特定できればよいので、徒にバイト数を多くし、若しくは数多く盛り込んで全体のデータ量を増やすべきではない。また、アプリケーションプログラムは、著作権者による許諾確認を行うシーケンスを持つことが必要である。

【００３５】

命令の種類が０ｘｆは、ＳＣＭ（Set Comment）命令であり、著作権情報設定命令である。ＳＣＭ命令は、これに続くオペランドにコメントを埋め込む。この具体例ではコメントに８バイトを割り当てているので、全体では７７ビットの命令になる。これもデータ量の増加に注意する必要がある。アプリケーションプログラムは、コメントを表示する手段を持つことが望ましい。このＳＣＭ命令に対して２５６バイトとか長いコメントを許容する場合には、そのコメントに対しても重ねてのデータ圧縮を図ることができる。その場合にはコメントの終了を示すためのコメント終了（Comment Termination）命令ＣＴを使う。本具体例の置き換え手段２がＳＣＭ命令を検出したときはそれ以後の文字列データをコメントとして扱い、ＣＴ命令を検出したときはコメントの終了を知る。

【００３６】

また、本具体例のデータ展開装置は、上記データ圧縮装置にて圧縮された出力データ１３を展開する装置である。即ち、生データとポインタと命令とを含むデータが入力された場合、この命令を実行して、生データまたはポインタに戻すと共に、生データまたはポインタを対象とするデータ列に戻すよう構成されたことを特徴とするデータ展開装置である。

【００３７】

〈動作〉

図６は、本発明のデータ圧縮装置の動作を示すフローチャートである。

先ず、入力データ１１に対して圧縮装置１が圧縮を行い、中間データ１２として、文字データとポインタを出力する。そして、これら文字データとポインタを

置き換え手段２に入力する（ステップＳ１）。これにより、置き換え手段２は、これら文字データとポインタを命令に置換できるか、あるいは特別の命令の追加を希望するかを判定する（ステップＳ２）。即ち、図５に示したような種々の命令に置換できるか、あるいは置換するかを判定する。

【００３８】

ステップＳ２において“Ｙ”の場合は、次にその命令が文字データ出力を伴うものであるかを判定する（ステップＳ３）。その命令が文字データ出力を伴わない場合（ステップＳ３で“Ｎ”）は、単にその命令を追加出力する（ステップＳ５）。一方、その命令が文字データ出力を伴う場合（ステップＳ３で“Ｙ”）は、それによりデータ量が少なくなるかをチェックする（ステップＳ４）。この判定は、命令に置き換えた場合と置き換えない場合の双方のデータ量を比較し、命令に置き換えた場合の方がデータ量が少なくなるかを調べることで行う。このステップＳ４において、データ量が少なくなると判定した場合は、ステップＳ５に進んでその命令による置換を行う。これにより、置き換え手段２からは、文字データ、ポインタ、命令が混在した出力データ１３が出力される。

【００３９】

尚、著作権情報の埋め込みを行う場合、置き換え手段２は、データ量が少なくなるかは考慮しないこととする。

【００４０】

図７は、命令への置換の説明図である。

先ず、図７の(1)は、同一の文字列を指すポインタを置き換える場合である。置き換え手段２が、第１のポインタＰ１と第２のポインタＰ２が同一の文字列Ｃを指すことを検出した場合、後のポインタＰ２を先のポインタＰ１を繰り返す命令ＲＰ（Ｂ，Ｌ）に置き換えようとする。また、置き換え手段２は、双方のコード長の総和を計算し、データ量が少なくなると判断した場合にその置き換えを行う。尚、この置き換えにより置き換えられてしまったポインタＰ２を操作する命令があった場合は、その命令が制限されてしまう。従って、本具体例の置き換え処理は、原則としてＬＺＳＳ等の出力データを先頭から順番に置き換えていくものでなければならない。このような置き換えるか否かの検出は、先ず、同一の文

字列を持つポインタか否かをチェックし、その後、文字列の同一性をチェックすることにより行う。

【００４１】

次に、図７の(2)は、隣り合う二つのポインタを置き換える場合である。隣り合う二つのポインタＰ１およびＰ２を、二つのポインタを統合する一つの命令ＣＰ（Ｂ１，Ｌ１，Ｂ２，Ｌ２）に置き換える。置き換え手段２は、ＬＺＳＳ等の出力データを先頭から順番にチェックし、隣り合う二つのポインタを発見したときは、それが本具体例で定義した命令ＣＰで置き換えられるか判断し、かつ、データ量が少なくなると判断した場合にのみその置き換えを行う。置き換えは、その隣り合う二つのポインタのコードを削除し、命令ＲＰに相当するコードを挿入することにより行う。ただ、当該ポインタを利用する他の命令等を考慮し、第２番目以降のポインタに適用するのが望ましい。

【００４２】

更に、図７の(3)は、１文字違いの文字列を指すポインタを命令で置き換える場合である。置き換え手段２が、文字列Ｃ１を指すポインタＰ１と１文字違いの文字列Ｃ２を指すポインタＰ２を検出したとき、後のポインタＰ２を先のポインタＰ１の文字列Ｃ１中のある１文字を変更する命令ＯＭＤ（Ｂ，Ｌ，Ｍ，Ｃ）に置き換えようとする。また、置き換え手段２が文字列Ｃ１を指すポインタＰ１を検出した後、これと１文字違いの文字列が存在し、これが複数の文字データやポインタで構成されていたとき、やはりその１文字を変更する命令ＯＭＤ（Ｂ，Ｌ，Ｍ，Ｃ）に置き換えようとする。置き換え手段２は、双方のコード長の総和を計算し、データ量が少なくなると判断したときにその置き換えを行う。この置き換えは、同一類似の文字列が多発するときに威力を発揮し、データ量の削減に寄与することができる。

【００４３】

図８は、本具体例による出力データの説明図である。

図８中の（ａ）が生データとして入力バッファにあるとする。この生データは、「ＡＢＣ」を６回繰り返し、その後「Ｄ…」という文字列が続くデータである。繰り返し部分の総ビット数は、１文字が１バイト、８ビットで構成されるので

、計１４４ビットとなる。

【００４４】

図８中の（ｂ）は、ＬＺＳＳにより（ａ）のデータをコード化した結果を示し、図１中の圧縮装置１の出力データに相当する。最初の文字データ列「ＡＢＣ」は２７ビット（（文字のビット数である８ビット＋特定のビットである１ビット）×３）であり、コード化のために１文字当たり１ビットだけ増加している。次の出力はポインタ「（３，３）」で、これはオフセットが３、長さが３なので、１１ビット（図３の拡張コード１０５を示す先頭ビットの１ビット＋オフセット１０６のためのビット数である８ビット＋長さ１０７を示す２ビット）である。また、同様に次に二つのポインタ「（６，６）（１２，６）」は、各々１３ビットであり、計６４ビットになる。従って、元のデータに対しておよそ４４％の圧縮がなされている。

【００４５】

図８中の（ｃ）は、具体例１の出力データである。文字データ列「ＡＢＣ」およびポインタ「（６，６）」は、ＬＺＳＳの場合と同様に行うが、次の出力データ「ＲＰ（１，４）」が異なっている。図５に示した定義表に示すように、この命令はポインタ制御命令であり、１個前のポインタを４回繰り返す命令である。１個前のポインタとは「（３，３）」であり、その出力データは文字データ列「ＡＢＣ」であるので、結局、文字データ列「ＡＢＣ」の出力が４回繰り返され、合計で５回になる。これは、当然であるが（ａ）に示した入力文字列と一致するコード内容を持っている。具体例１では、図５に示したようにビット数を割り当てたので、このポインタ制御命令は２４ビットで構成できる。その結果、出力データは計６２ビットで、これは元のデータに対しておよそ４３％の圧縮を達成することができる。

【００４６】

図８中の（ｄ）は具体例１による他の命令を用いた出力データである。文字データ列「ＡＢＣ」は、ＬＺＳＳの場合と同様に行うが、次の出力データ「ＲＤ（３，３，５）」が異なっている。図５に示した定義表に示すように、この命令はデータ制御命令であり、３個前の文字から３個の文字列を５回繰り返す命令であ

る。即ち、この命令により文字データ列「ＡＢＣ」を５回出力する。本具体例では、各パラメータについて図５のようにビット数を割り当てたので、このポインタ制御命令は３２ビットで構成できる。結局、出力データは計５９ビットで、これは元のデータに対しておよそ４１％の圧縮を達成することができる。

【００４７】

このように、本具体例では、図８に示すようなデータに対して、ＬＺＳＳ等のデータ圧縮方式に比べて更に数％圧縮率を向上させることができる。

【００４８】

次に、このような本具体例のデータ圧縮装置によって圧縮されたデータを展開するデータ展開装置（デコーダ）の動作を説明する。

図９は、データ展開装置の動作を示すフローチャートである。

先ず、文字データ、ポインタおよび命令で構成されるコードの一つを入力する（ステップＳ１１）。先ず、そのコードが文字データであるかを判定し（ステップＳ１２）、そうであった場合はその文字を出力し（ステップＳ１３）、次のコードに移る。ステップＳ１２において、文字データではなかった場合は、ポインタであるかを判定し（ステップＳ１４）、そうであった場合は、そのポインタの示す文字列の出力を行い（ステップＳ１５）、次のコードに移る。ステップＳ１４において、そのコードがポインタではなかった場合は、命令であるかを判定する（ステップＳ１６）。即ち、そのコードが終端マーカ２１で始まるコードであるかを判定する。ここで、本具体例の場合、終端マーカで始まるコードは命令と判断することができる。

【００４９】

ステップＳ１６において、そのコードが命令でもない場合（ステップＳ１６で“Ｎ”）は、エラー表示を行い（ステップＳ１７）、その時点でデータ展開処理を終了する。即ち、コードが、文字データ、ポインタ、命令のいずれでも無い場合は存在しないため、データ展開処理を中止する。

【００５０】

ステップＳ１６においてコードが命令であり、かつ、その命令がデータ出力命令である場合（ステップＳ１８において“Ｙ”）は、その命令を解釈して該当す

る文字列を出力し（ステップＳ１９）、次のコードに移る。ステップＳ１８において、命令の種類がデータ出力命令ではない場合（ステップＳ１８において“Ｎ”）は、その命令が終端マーカ（図５中のＥＮＤ命令）であるかを判定する（ステップＳ２０）。ステップＳ２０において、命令が終端マーカを示すものであった場合は、展開対象となるデータが最後であるため、データ展開処理を終了する。一方、ステップＳ２０において、命令が終端マーカではない場合は、パスワード設定命令ＳＰＷや著作権情報設定命令ＳＣＲやコメントＳＣＭであるため、著作権情報の表示等、対応した処理を行い（ステップＳ２１）、次のコードに移る。

【００５１】

〈効果〉

以上のように、具体例１によれば、命令が対象とする文字列が、文字データの対象とする文字列とポインタが指示する文字列、または、これらの組合せと一致する場合に、文字データまたはポインタに置き換えて命令を出力するようにしたので、例えば、著作権情報やパスワードといった種々の情報も容易に埋め込むことができ、自由な圧縮が可能となる。また、圧縮率の向上に寄与することができる。

【００５２】

また、具体例１においては、命令に割り当てられたコード長が、文字データとポインタとからなるデータ長よりも少ない場合、その元となる文字データまたはポインタを命令に置き換えるようにしたので、本具体例の出力データが従来のＬＺＳＳ等の圧縮方式による出力データよりも圧縮率が低下してしまうことを防止することができる。

【００５３】

《具体例２》

具体例２は、命令の識別を、特定のビット列で行うようにしたものである。

【００５４】

〈構成〉

本具体例のデータ圧縮装置の図面上の構成は、図１と同様であるため、ここで

の説明は省略する。本具体例の置き換え手段２は、そのコード化の構成が具体例１とは異なっている。

【００５５】

図１０は、具体例２のコード化の説明図である。

先ず、出力データ（Compressed Stream）２０１は、圧縮されたデータ列（Compressed String）２０２と終端マーカ（End Marker）２０３から構成される。圧縮されたデータ列２０２は、先頭のビットが０＋生データ（Raw Byte）２０４か、若しくは、先頭のビットが１＋命令（Command/Pointer）２０５の集合である。尚、この「先頭のビット」とは、上述した特定のビット（an extra ID-bit）のことである。ここで、生データ２０４は、ＡＳＣＩＩのような８ビット（１バイト）で構成されるが、命令（Command/Pointer）２０５では、ポインタ（Pointer）２０７も一種の命令であると考え、先頭ビットが０のポインタ（Pointer）２０７と先頭ビットが１の命令（Command）２０６とを識別する。この識別は、本具体例で命令を優先するコード化を採用したため導入されたものである。但し、この識別のために、１ビット分コード長が長くなる。

【００５６】

ポインタ（Pointer）２０７はオフセット（Offset）２１０とコード長（Length）２１１で構成されている。オフセット２１０は、コード化の効率を考慮して、先頭ビットが１である６ビットオフセット（合計７ビット）、若しくは先頭ビットが０である１１ビットオフセット（合計１２ビット）を用意している。具体例１のコード方式では、先頭ビットが１である７ビットオフセットを採用していたが、命令とポインタの識別のために、上述したように１ビット分だけコード長が長くなっているので、これを補う意味でオフセットを短くしている。また、コード長２１１は、具体例１と同様に、出現頻度の高い短いデータ長のポインタに短いコード長のコードを割り当てる統計型圧縮方式である。

【００５７】

本具体例の命令（Command）２０６は、命令の種類（Command Set）２０８とオペランド（Operand）２０９から構成され、本具体例では４ビットを命令の種類に割り当てているため、１６命令を指定することができる。オペランド２０９は

その命令のパラメータを指定するのに使用する。また、本具体例では、終端マーカ（End Marker）２０３は命令の一つであり、具体的にはコード「１１０００」という６ビットで定義されている。

【００５８】

本具体例は、具体例１と比べると、命令に対して短いコードを割り当てたところにその特徴がある。即ち、圧縮されたデータ列２０２の先頭の２ビットが“１１”であれば、命令と判断される。生データ２０４のビット数は具体例１と同様９ビットであるが、ポインタ２０７は１ビット分長くなり、その分重要度が低下している。但し、その分総ビット数が短くなった命令によりデータ圧縮に寄与することができる。

【００５９】

図１１は、具体例２の命令の構成を示す説明図である。

本具体例において、図４に示した具体例１の構成と異なる点は、命令のコードを具体例１の終端マーカ２１から、短い命令コード「１１」３１に短縮したことである。即ち、具体例１では、図３の終端マーカ１０３に０ｘ１８０というコードを割り当て、これに命令の種類等のオペランドを追加していたのに対し、この具体例２では、全ての命令の先頭に短い命令コード「１１」を割り当て、更に、命令の種類に応じて４ビットおよびパラメータに相当するビット数を割り当てている。これは、本発明で導入した「命令」に対し、ポインタと比較して、より優先的な地位を与えたことを意味する。本具体例の命令を活用すれば、ポインタしかなかった従来のＬＺＳＳ等の圧縮方式に比べて格段の圧縮率を達成することができる。また、本具体例では、拡張コード３２（命令の種類３２ａ＋オペランド３２ｂ）は、図１１（ｂ）に示すように、図４の命令の種類２２ａ＋オペランド２２ｂと同様に設定されている。

【００６０】

また、命令の種類は、具体例１で説明した図５の定義表と同様に作成することができる。

図１２は、具体例２における命令の作成例の説明図である。

ここで、図示のように具体例１と異なるのは、具体例１の終端マーカ０ｘ１８

０に代わってコード「１１」が割り当てられている点であり、その他の点については同様である。

【００６１】

〈動作〉

具体例２のデータ圧縮装置におけるデータ圧縮動作は図６に示した具体例１の動作の流れと同様であるため、ここでの説明は省略する。また、その一例の動作を説明するため、図８を援用して説明する。

【００６２】

先ず、図８（ｃ）で、先頭の文字列「ＡＢＣ」は、図１０に示したコード化を行うと、従来のコード化と同じであるので、１文字当たり９ビット、合計で２７ビット必要となる。次のポインタ「（３，３）」は、具体例１ではコード「１１０００００１１０１」であったが、具体例２ではコード「１０１０００００１１０１」であって同じく１１ビットで構成できる。これは、短い方のオフセットを具体例２では６ビットにしたからである。次の出力データ「ＲＰ（１，４）」は、本具体例による命令であって、具体例１では２４ビットであったが、図１０のコード化に従えば「１１００１０００００００１０１００」であって、１７ビットで構成できる。従って、合計のビット数は５５ビットであり、圧縮前の１４４ビットと比較すると３８％であって、ＬＺＳＳの４４％よりもかなり圧縮率が向上している。

【００６３】

次に、図８（ｄ）で、同様に出力データ「ＲＤ（３，３，５）」をコード化すると、これは具体例３の場合は２５ビットで表現でき、合計のビット数は５２ビットであり、圧縮前の１４４ビットと比較すると３６％であって、ＬＺＳＳの４４％よりも一段と圧縮率が向上している。

【００６４】

尚、データ展開処理は、命令の判定が先頭２ビットが“１１”であるか否かに基づいて行う点が具体例１と異なるだけで、他の動作は同様であるため、ここでの説明は省略する。

【００６５】

〈効果〉

以上のように、具体例２によれば、第１のビットを文字データとポインタと命令とを分別することに用い、第２のビットをポインタと命令とを分別することに用いるようにしたので、具体例１に比べて命令に関するコードが短くて済み、更に圧縮率を向上させることができる。

【００６６】

《具体例３》

具体例３は、動的コード割り当て（Dynamic Code Assignment）方式と呼ぶ新しいコード化の方式に関する。この方式は、統計型圧縮方式（the entropy coding methods）と比べると、文字列や命令等に割り当てたコードが動的に変化する点で、従来の圧縮方式とは異なるものである。統計型圧縮方式では、短い文字列ほど頻繁に出現するので、これに短いコードを割り当てようとするものであり、これは、いわば経験則に基づくコード化方式である。

【００６７】

これに対し、具体例３の動的コード割り当て方式は、入力バッファといった所定のデータ量の定義単位毎に、その中で実際に頻繁に出現する文字列をコード化する方式であって、定義単位毎に異なるコード化を行うことから、出力データの中の同じコードが異なる意味を持つ現象が発生する。この方法は、実際に頻繁に出現する文字列に短いコードを割り当てようとするので、圧縮率を更に高めることができる。

【００６８】

〈構成〉

図面上の構成は、具体例１における図１と同様であるため、ここでの図示は省略する。具体例３のデータ圧縮装置が具体例１、２と異なるのは、置き換え手段２がコード化処理する単位が、例えば入力バッファ単位といった所定のデータ量の定義単位となっている点である。また、具体例３では、文字列や命令に対して統計型圧縮方式を適用している。

【００６９】

図１３は、具体例３における命令の作成例を示す説明図である。

具体例３の基礎は、データ定義（Data Definition）命令と、コード置き換え（Code Substitution）命令にある。図１３に示すように、データ定義命令として文字列定義命令ＳＤ（String Definition）、ＳＤＯ（String Definition and Out）およびポインタ定義命令ＰＤ（Pointer Definition）を用意している。ＳＤ（Ｂ，Ｌ，Ｍ）命令は、Ｂ個前からＬ個の文字列をＭ番と定義する。ここで、「Ｍ番」は、後述するように、その文字列の入力バッファ内の出現頻度の順位とするのが一般的である。ＳＤＯ（Ｂ，Ｌ，Ｍ）命令は、Ｂ個前からＬ個の文字列をＭ番と定義し、かつ、出力するものである。この命令の存在意義は、文字列の定義とポインタの機能を同時に一つのコードで達成するので、その分コード量を節約することができる点にある。ＰＤ（Ｍ）命令は、直前のポインタをＭ番と定義するものである。同様に、「Ｍ番」はその文字列の出現頻度の順位にするのが一般的である。また、コード置き換え命令としてはＣＳ（Ｍ）命令を示した。このＣＳ（Ｍ）命令は、データ定義命令で指定したＭ番の文字列等を短いコードに置き換える命令である。

【００７０】

本具体例の特徴は、このコード置き換え命令ＣＳ（Ｍ）自体のコード長を非常に短くしておき、かつ、出力バッファ中の出現頻度の順位に従って文字列や命令等をコードに置き換えれば、全体として高い圧縮率を達成できるというものである。また、入力バッファ毎に、このような定義化を行うことで、入力データの局所的なデータ構造を反映した定義化が行われ、従って、従来のように、入力データの全体に亘って固定したコード化を行う圧縮方式に比べて、より適切な圧縮を行うことができる。

【００７１】

図１４は、ある入力バッファでの同一の文字列の出現頻度とコード化との関係を示す説明図である。

具体例３の動的コード割り当て方式を適用する場合に、データ定義命令で指定する文字列は、その命令を構成するパラメータＭのビット数により制限される。本具体例ではＭは５ビットであり、３２個の文字列を指定することが可能である。具体例１、２において、入力バッファは２Ｋバイト（オフセットが最大１１ビ

ットであることによる値）であり、本具体例においても、定義単位である入力バッファは２Ｋバイトであるとする。このような入力バッファにおいて、パラメータＭのビット数を最適化するには、実験的に求める必要があるが、本具体例では５ビットの値としている。

【００７２】

図１４（ａ）に示すように、出現頻度（Ｔ）が高くてもパラメータ（Ｍ）のビット数による制限から定義可能な文字列の最大数は３２個である。そこで、順位Ｍが３２以下の出現頻度を有する文字列の定義等は行わず、パラメータＭのビット数を制限として定義の打ち切りを行う。また、図１４（ｂ）は、出現頻度（Ｔ）による打ち切りを行う例である。これは、ＬＺＳＳにおいてマッチ（Match）の長さを通常２バイトにする理由と似ている。即ち、出現頻度（Ｔ）の低い文字列を定義し、かつ、置き換えても圧縮率向上への寄与は小さいからである。そこで、図示例では、出現頻度（Ｔ）が２以下の場合は、定義等を行わないようにしている。尚、ある入力バッファでの同一の文字列を指定する複数のポインタはそれぞれオフセット値が異なるが、図１４で示した例では、同一の文字列を指定する同一のポインタとしてその出現頻度を数えている。

【００７３】

同一のポインタか否かの判定は、入力バッファ内の現在対象としている文字列のアドレスとオフセット値との関係から定まる。

図１５は、同一のポインタか否かの判定手段を示す説明図である。

図示例では、ポインタＰ２がポインタＰ１と同一のポインタか否かを判定する場合を示している。判定条件の第一は、二つのポインタＰ１とＰ２が指し示す文字列が同一の文字数を示すことであり、それは文字列Ｓ１の文字数と一致する。判定条件第二は、二つのポインタＰ１とＰ２が入力バッファ内で持つ相対アドレスPadrとCadrとの差が、二つのポインタＰ１とＰ２が有するオフセット値OffsetadrPとOffsetadrCの差に一致することである。尚、ここでは、現在検査対象としているポインタがＰ２であることを示すために、「Ｃ（Current）」の文字を用いている。

【００７４】

図１６は、文字列等の定義表の作成例を示す説明図である。

この定義表は、入力バッファ毎に、置き換え手段２が作成するものである。指定順位Mに存在する文字列を定義する定義命令がそのパラメータと共にリストアップされている。ここで、各指定順位の定義命令は、それぞれが異なる文字列やポインタを示している。例えば、指定順位１の定義命令ＳＤＯ（Ｂ，Ｌ，Ｍ）と指定順位４の定義命令ＳＤＯ（Ｂ，Ｌ，Ｍ）とは、異なる文字列を表している。有効／無効フラグは、対応する定義が有効か無効かを示すフラグであり、本具体例では、具体例２における図１１（ｂ）に合わせて、指定順位が１６位迄が有効（値が１である）な定義となっている。尚、定義表はアプリケーションプログラムが利用できるメモリ内に置くことができる。この場合に、指定順位Mが重複しないように処理したコードを記憶しておけばメモリ量を減らすことができる。

【００７５】

図１７は、具体例３におけるコード化の一例を示す説明図である。

本具体例のコード化の手法は、具体例２における図１０のコード化と対比することができる。

出力データ（Compressed Stream）３０１は、圧縮されたデータ列（Compressed String）３０２と終端マーカ（End Marker）３０３から構成される。圧縮されたデータ列３０２は、先頭のビットが０＋生データ（Raw Byte）３０４か、若しくは、先頭のビットが１＋命令（Command/Pointer）３０５の集合である。ここで、生データ３０４は、ＡＳＣＩＩのような８ビット（１バイト）で構成されるが、命令（Command/Pointer）３０５では、ポインタ（Pointer）３０７も一種の命令であると考え、先頭ビットが０のポインタ（Pointer）３０７と先頭ビットが１の命令（Command）３０６とを識別する。

【００７６】

ポインタ（Pointer）３０７はオフセット（Offset）３１０とコード長（Length）３１１で構成されている。オフセット３１０は、コード化の効率を考慮して、先頭ビットが１である６ビットオフセット（合計７ビット）、若しくは先頭ビットが０である１１ビットオフセット（合計１２ビット）を用意している。コード長３１１は、具体例１、２と同様に、出現頻度の高い短いデータ長のポインタ

に短いコード長のコードを割り当てる統計型圧縮方式である。

【００７７】

本具体例の特徴は、主に命令のコード化にある。本具体例の命令（Command）３０６は、命令の種類（Command Set）３０８とオペランド（Operand）３０９から構成される。そして、命令の種類をコード化するにあたって、統計型圧縮方式を導入している。その理由は、命令の使用頻度がかなりばらつくと考えられるからである。例えば、ＣＳ命令にはコード「００」を割り当てている（図中のCommand Set３１２中の３１４）。これは、ポインタ等の置き換えのために最も頻繁に出現すると考えられるからである。次に、ＳＤＯ命令にはコード「０１」を割り当てた（図中、３１５）。定義命令は通常指定順位Ｍの最大値まで利用されるから出現頻度が高い。次に、ＰＤ命令はコード「１０」を割り当てた（図中、３１６）。この命令もポインタの代用として利用される可能性が高い。尚、どの命令も頻繁に出現する場合には、単純に４ビットを割り当てる図１０に示した具体例２の方式を用いることもできる。

【００７８】

また、本具体例では、２２種類の命令を定義し、ＥＮＤ命令にはコード「１１１１１１１」を割り当てた（図中、３１７）。また、オペランド３０９についても、統計型圧縮方式を導入することができる（図中、３１３）。指定順位に割り当てたパラメータＭは、順位の高い程短いコードにすることが望ましい（図中、３１８に示す）。ここでは第２２位までの指定を確保している。その他のパラメータについても統計型圧縮方式を採用することができる。尚、ポインタの文字数をコード化する際に適用した統計型圧縮方式とは若干異なる方式となっている。これは、ＬＺＳＳではポインタ長が２バイトにしてあることから最小の長さが２だからであり、本具体例では命令コードやパラメータにそのような制限を付ける必要がないからである。

【００７９】

〈動作〉

図１８は、具体例３における置き換え手段２の動作を示すフローチャートである。

先ず、処理が開始され、入力バッファのデータ更新が行われる（ステップＳ３１）と、定義表の初期化を行う（ステップＳ３２）。定義表の初期化は、図１６に示した有効／無効フラグを無効にする（値０を書き込む）ことにより行う。次に、圧縮装置１によって、入力バッファにある生データが圧縮処理され（ステップＳ３３）、中間圧縮出力として文字データやポインタが出力される（ステップＳ３４）。即ち、このステップＳ３３、Ｓ３４の圧縮処理は、従来のＬＺＳＳによる圧縮処理に相当する。

【００８０】

次に、置き換え手段２は定義表への書き込みを行う（ステップＳ３５）。即ち、対象となる入力バッファ内の文字列の出現頻度を求め、出現頻度の高いもの程、指定順位（Ｍ）を上げるように、定義表への書き込みを行う。即ち、図１６において、それぞれの指定順位（Ｍ）の文字列に対応した定義命令のコードを書き込み、かつ、有効／無効フラグを有効（値１を書き込む）にする。そして、置き換え手段２は、各種圧縮処理を行う（ステップＳ３６）。この各種圧縮処理は、例えば、図７で示したようなポインタの置換処理、著作権情報等の埋め込み処理、更にコード置き換え命令の設定といった処理である。

【００８１】

図１９は、コード置き換え命令の設定の説明図である。

図１９（ａ）は、一つのポインタのみを取り扱う場合で、中間データ１２である文字データ等として、「．．．．Ｐ１．．．Ｐ１．．Ｐ１．．．．．Ｐ１．．．」があり、置き換え手段２を用いて、出力データ１３として、「．．．．ＳＤＯ．．．ＣＳ．．ＣＳ．．．．．ＣＳ．．．」が出力されている。この具体例では、最初のポインタ「Ｐ１」を文字列定義出力命令ＳＤＯに置き換えている（図中、①参照）。尚、ポインタ定義命令ＰＤを使うとポインタ一つ分重複する。しかしその場合にはそのポインタを操作する命令を使用することができるため、最終的な圧縮率を基準に判断することが必要である。第２番目以降のポインタについてはコード置き換え命令ＣＳに置き換えている（図中、②～④参照）。尚、各々のポインタＰ１はオフセットが異なるが、ここでは同じ文字列を指し示すポインタとして扱っている。

【００８２】

図１９（ｂ）は、複数のポインタを取り扱う場合、中間データ１２中の最初のポインタ「Ｐ１」および「Ｐ２」を、出力データ１３では、それぞれ文字列定義出力命令「ＳＤＯ１」および「ＳＤＯ２」に置き換え、それ以降のポインタ「Ｐ１」および「Ｐ２」をそれぞれコード置き換え命令「ＣＳ１」および「ＣＳ２」に置き換えている。このように、複数のポインタを取り扱う場合でも、各ポインタはそれぞれ独立に定義し、置き換えることができる。

【００８３】

図１８に戻って、データが終了したかを判定し（ステップＳ３７）、また、処理すべき入力バッファのデータがあった場合は、上述したステップＳ３１からの処理を繰り返す。全てのデータが終了した場合は、データ圧縮処理を終了する。

【００８４】

次に、本具体例のデータ圧縮装置で圧縮された出力データをデコーダがどのように扱うかを説明する。

ここで、デコーダの動作は、基本的には図９の動作と同様であるため、図９を援用して説明する。具体例３で異なるのは、データ出力命令の扱いである。例えば、データ出力命令以外の命令では、具体例１、２の場合、著作権情報表示等を行う（ステップＳ２１）扱いになっているが、本具体例では、データ定義命令（例えば、ＳＤＯ命令）の場合、図１６に示したような定義表への書き込みを行う。即ち、入力バッファといった所定のデータ単位毎に定義表を作り直すことになる。また、コード置き換え命令（例えば、ＣＳ命令）については、データ出力命令として扱う（図９において、ステップＳ１８で“Ｙ”）。即ち、そのコード置き換え命令のパラメータである指定番号（Ｍ）に相当するデータ定義命令を定義表で参照し、その定義されている文字列を出力する。この場合、その指定番号Ｍの有効／無効フラグを参照してそれが無効であった場合にはデコーダエラーとなる。

【００８５】

尚、具体例３では、同一の文字列を指定する複数のポインタであって、そのオフセットが異なるものの出現頻度を基準にして指定順位Ｍを決定した。しかし、

これ以外の基準を採用することもできる。例えば、文字列数とポインタ数の積をデータ総量と考え、これを基準にすることも考えられる。しかし、通常短いデータほど出現頻度が高いため、この基準では圧縮率への寄与が少ない場合が発生する。また、文字列のコード数とポインタ数の積をコード総量と考え、これを基準にすることが考えられる。しかし、ポインタ数が少ないときには圧縮の効果も少ないことが考えられる。

【００８６】

〈効果〉

以上のように、具体例３によれば、特定のポインタが指定する文字列と、他の複数のポインタが指定する文字列とが一致する場合に、特定のポインタを定義命令に置き換え、かつ、他の複数のポインタを定義命令に対応するコード置き換え命令に置き換える処理を、入力バッファといった所定のデータ単位毎に行うようにしたので、実際に頻繁に出現する文字列に動的に短いコードを割り当てることができ、その結果、データ構造の局所的構造を的確に捉え圧縮率を更に向上させることができる。

【００８７】

また、具体例３によれば、複数の命令が、それぞれ命令の種類とオペランドにより構成される場合に、各命令の種類や各オペランドのパラメータに応じて漸次コード長を増加させるコード化を行うようにしたので、頻繁に出現する命令に短いコードを割り当てることができ、その結果、更に圧縮率を向上させることができる。

【００８８】

また、具体例３によれば、特定のポインタが指定する文字列として、入力バッファ等の所定のデータ単位内に存在する文字列のうち、最初に出現したものを選択するようにしたので、定義命令への置き換えやコード置き換え命令への置き換えを容易に行うことができる。

【００８９】

また、具体例３によれば、複数の定義命令を設定した場合に、これらの定義命令に置き換えたポインタが指示していた複数の文字列の、入力バッファといった

所定のデータ単位内での出現頻度を計数し、その出現頻度の順番に、定義命令を記載した定義表を作成するようにしたので、実際に頻繁に出現する文字列に動的に短いコードを割り当てることができ、従って、圧縮率を向上させることができる。

【００９０】

また、具体例３によれば、特定のポインタが指定する文字列の文字数と、他のポインタが指定する文字列の文字数とが一致するかを判定し、かつ、入力バッファといった所定のデータ単位における特定のポインタの指定する文字列のアドレスと、他のポインタの指定する文字列のアドレスとの差が、特定のポインタの持つオフセット値と、他のポインタの持つオフセット値との差に一致するかを判定し、一致した場合に、特定のポインタが指定する文字列と、他のポインタが指定する文字列とが一致すると判定するようにしたので、異なるポインタが指定する文字列が一致するか否かを容易に判定することができる。

【００９１】

尚、上記各具体例において、置き換え手段２に入力される中間データ１２として、従来のＬＺＳＳ等による文字データやポインタであるとしたが、圧縮装置１の圧縮方式はＬＺＳＳに限定されるものではなく、置き換え手段２への入力として、文字データ等の生データとポインタとからなるデータであれば、中間データ１２はどのようなデータであってもよい。

【００９２】

【発明の効果】

以上のように、本発明によれば、命令が対象とするデータ列が、生データの対象とするデータ列とポインタが指示するデータ列、または、これらの組合せと一致する場合に、生データまたはポインタに置き換えて命令を出力するようにしたので、例えば、著作権情報やパスワードといった種々の情報も容易に埋め込むことができ、自由な圧縮が可能となる。また、圧縮率の向上に寄与することができる。

【図面の簡単な説明】

【図１】

本発明の具体例１におけるデータ圧縮装置の構成図である。

【図２】

ＬＺＳＳのコード化の説明図である。

【図３】

ＬＺＳＳ等で利用されるコード化の説明図である。

【図４】

具体例１における命令の基本構成を示す説明図である。

【図５】

具体例１における命令の作成例を示す説明図である。

【図６】

具体例１のデータ圧縮装置の動作を示すフローチャートである。

【図７】

命令への置換の説明図である。

【図８】

具体例１の出力データの説明図である。

【図９】

具体例１のデータ展開装置の動作を示すフローチャートである。

【図１０】

具体例２のコード化の説明図である。

【図１１】

具体例２の命令の構成を示す説明図である。

【図１２】

具体例２における命令の作成例の説明図である。

【図１３】

具体例３における命令の作成例を示す説明図である。

【図１４】

ある入力バッファでの同一の文字列の出現頻度とコード化との関係を示す説明図である。

【図１５】

同一のポインタか否かの判定手段を示す説明図である。

【図１６】

文字列等の定義表の作成例を示す説明図である。

【図１７】

具体例３におけるコード化の一例を示す説明図である。

【図１８】

具体例３の動作を示すフローチャートである。

【図１９】

コード置き換え命令の設定の説明図である。

【符号の説明】

１　圧縮装置

２　置き換え手段

１１　入力データ

１２　中間データ

１３　出力データ

２２、３２　拡張コード

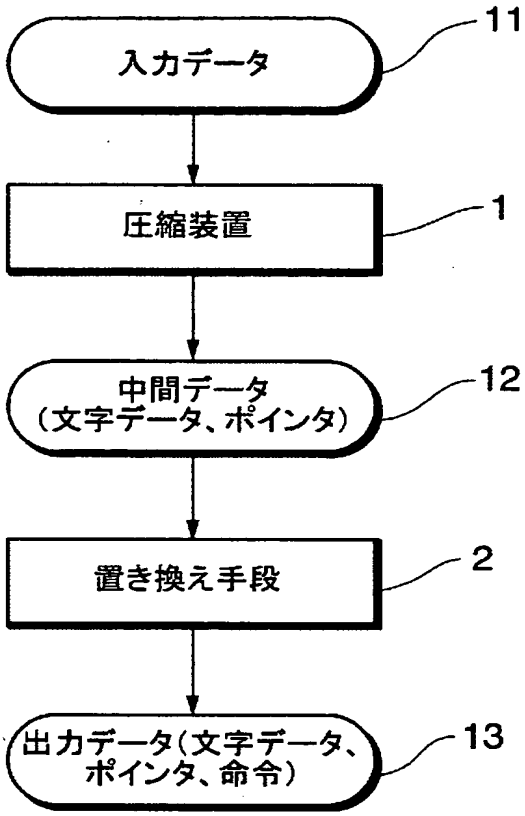
２２ａ、３２ａ　命令の種類

２２ｂ、３２ｂ　オペランド

３１　命令コード

【書類名】図面

【図１】



本発明のデータ圧縮装置の構成図

【図２】

Input stream for encoding:

Pos 1 2 3 4 5 6 7 8 9 10 11

Char A A B B C B B A A B C

Table 1:

The encoding process

(MIN_LENGTH=2) Step	Pos	Match	Output
1	1	–	A
2	2	A	A
3	3	–	B
4	4	B	B
5	5	–	C
6	6	BB	(3,2)
7	8	AAB	(7,3)
8	11	C	C

LZSSによるコード化の説明図

【図３】

<Compressed Stream> := [<Compressed String>] <End Marker>
101 102 103

<Compressed String> := 0 <Raw Byte> |1 <Compressed Bytes>
104 105

<Raw Byte> := <b><b><b><b><b><b><b><b> (8-bit byte)
<Compressed Bytes> := <Offset><Length>
106 107

<Offset> :=1 <b><b><b><b><b><b><b> (7-bit offset)
0 <b><b><b><b><b><b><b><b> <b><b><b> (11-bit offset)

<End Marker> := 110000000

<b> := 1 | 0

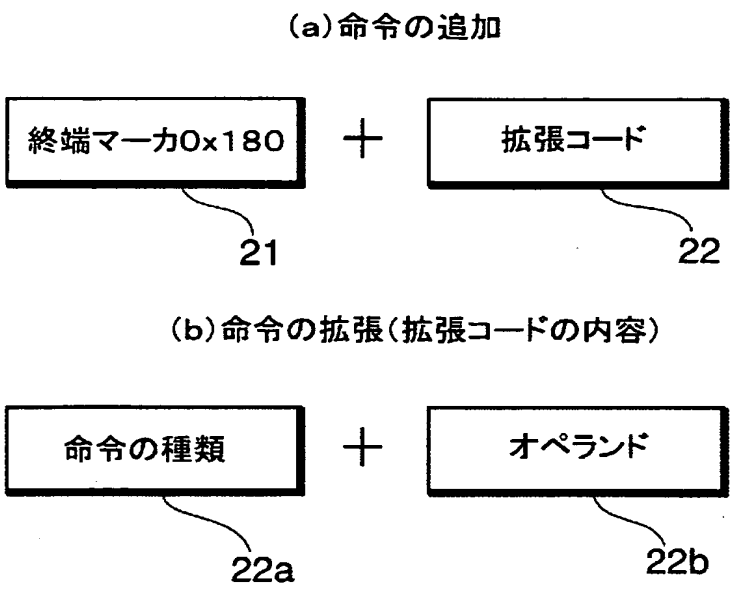
<Length> :=

00 = 2	1111 0110 = 14
01 = 3	1111 0111 = 15
10 = 4	1111 1000 = 16
1100 = 5	1111 1001 = 17
1101 = 6	1111 1010 = 18
1110 = 7	1111 1011 = 19
1111 0000 = 8	1111 1100 = 20
1111 0001 = 9	1111 1101 = 21
1111 0010 = 10	1111 1110 = 22
1111 0011 = 11	1111 1111 0000 = 23
1111 0100 = 12	1111 1111 0001 = 24

1111 0101 = 13 ・・・

コード化の説明図

【図４】



具体例1の命令の基本構成の説明図

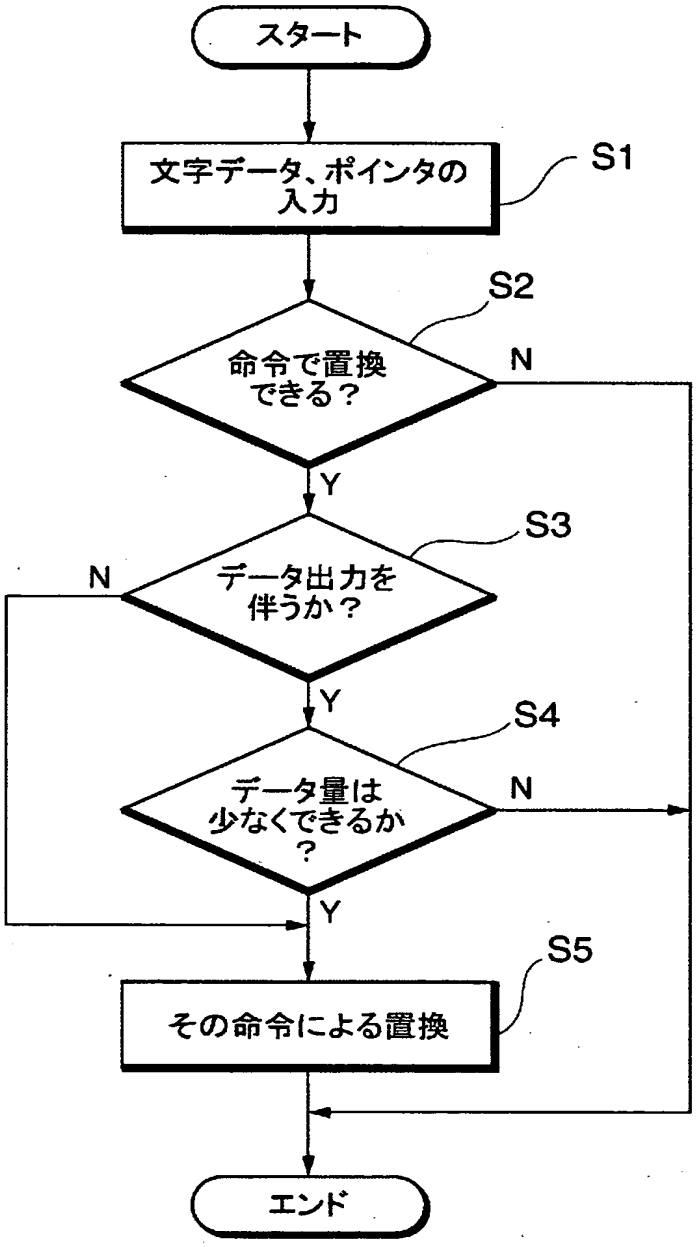
【図５】

命令の種類	コード	意義
END 終端マーカ	0x180 + 0x0	出力データの終わり
RD(B, L, N) データ制御命令	0x180 + 0x1 + (B,L,N)	B個前からL個の文字データ列を N回繰り返す
RP(B, N) ポインタ制御命令	0x180 + 0x2 + (B,L)	B個前ポインタをN回繰り返す
RDP(B, L, N) 出力データ列制御命令	0x180 + 0x3 + (B,L,N)	B個前からL個の出力データ列を N回繰り返す
OMD(B, L, M, C) データ制御命令	0x180 + 0x4 + (B,L,M,C)	B個前L個の文字データ列のM番目の データをCに変更し出力する
CP(B1, B2) ポインタ制御命令	0x180 + 0x5 + (B1,B2)	ポインタP1とP2が指す文字データ列を 結合して出力する
⋮	⋮	⋮
SPW パスワード設定命令	0x180 + 0xD + (PW)	パスワードPWを設定する
SCR 著作権情報設定命令	0x180 + 0xE + (CR)	著作権情報CRを設定する
SCMコメント	0x180 + 0xF + (CM)	コメントCMを挿入する

但し、B:7ビット　N:4ビット　L:8ビット　M:8ビット　C:8ビット　とする。

具体例1の命令の作成例の説明図

【図６】



具体例１の動作フローチャート

(1)隣り合う2つのポインタの置換

隣り合う2つのポインタP1及びP2を、2つのポインタを統合する1つの命令CP(B1, L1, B2, L2)に置き換える。

「....P1P2....」→「....CP....」

(2)同一の文字列を指すポインタの置換

同一の文字列Cを指すポインタP1及びP2を検出したとき、後のポインタP2を先のポインタP1を繰り返す命令RP(B. L)に置き換える。

「....C..P1..P2....」→「....C..P1..RP....」

(3)1文字違いの文字列を指すポインタの置換

文字列C1を指すポインタP1と1文字違いの文字列C2を指すポインタP2を検出したとき、後のポインタP2を先のポインタPlの文字列C1の当該1文字を変更する命令OMD(B, L, M, C)に置き換える.

「....C1...C2..P1..P2....」→「....C1...C2..P1..OMD....」

命令への置換の説明図

【図８】

(a)入力文字列

ABC ABC ABC ABC ABC ABC D.....
(24) (24) (24) (24) (24) (24) 計144ビット

(b)LZSSによる出力データ

ABC (3,3) (6,6) (12,6) D.....
(27) (11) (13) (13) 計64ビット

(c)本発明の具体例による出力データ

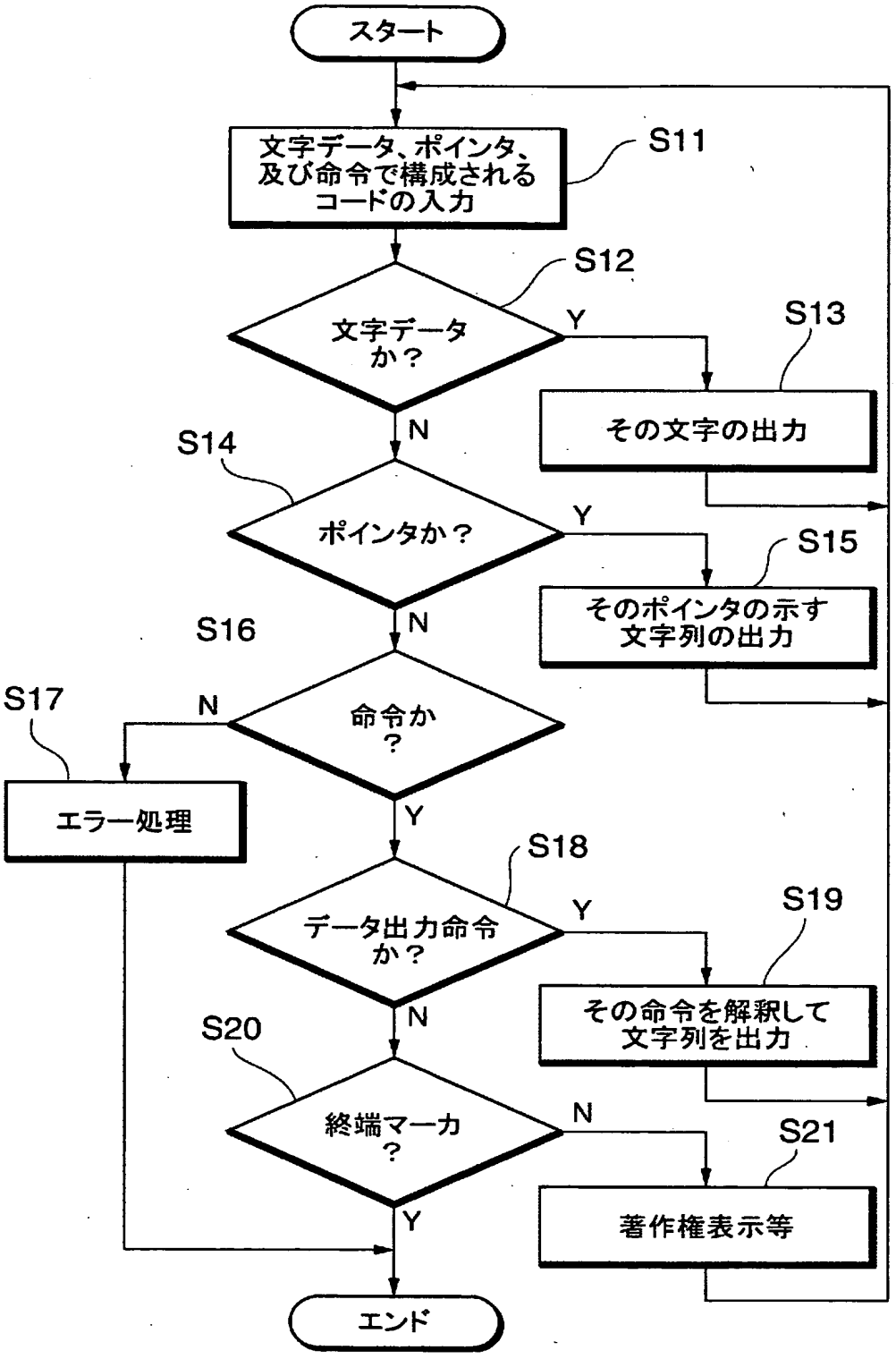
ABC (3,3) RP(1,4) D.....
(24) (11) (24) 計62ビット

(d)本発明の具体例による出力データ

ABC RD(3,3,5) D.....
(27) (32) 計59ビット

具体例1の出力データの説明図

【図９】



具体例1のデータ展開装置の動作フローチャート

【図１０】

<Compressed Stream> := [<Compressed String>] <End Marker>
201 202 203
<Compressed String> := 0 <Raw Byte> |1 <Command/Pointer>
204 205
<Raw Byte> := <b><b><b><b><b><b><b><b> (8-bit byte)

<Command/Pointer> := 1 <Command> | 0 <Pointer>
206 207
<Command> := <Command Set><Operand>
208 209
<Command Set> := <b><b><b><b> (4-bit>
<End Marker> := 110000

<Pointer> := <Offset><Length>
210 211
<Offset> :=
1<b><b><b><b><b><b><b> (6-bit offset)
0 <b><b><b><b><b><b><b><b> <b><b><b> (11-bit offset)

<b> := 1 | 0

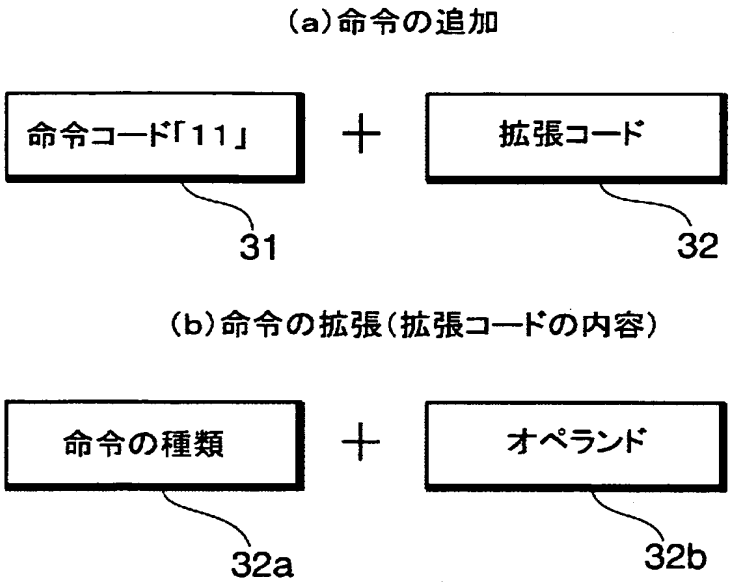
<Length> :=

00 = 2	1111 0110 = 14
01 = 3	1111 0111 = 15
10 = 4	1111 1000 = 16
1100 = 5	1111 1001 = 17
1101 = 6	1111 1010 = 18
1110 = 7	1111 1011 = 19
1111 0000 = 8	1111 1100 = 20
1111 0001 = 9	1111 1101 = 21
1111 0010 = 10	1111 1110 = 22
1111 0011 = 11	1111 1111 0000 = 23
1111 0100 = 12	1111 1111 0001 = 24

1111 0101 = 13 …

具体例2のコード化の説明図

【図１１】



具体例2の命令の基本構成の説明図

命令の種類	コード	意義
END 終端マーカ	「11」＋0x0	出力データの終わり
RD(B, L, N) データ制御命令	「11」＋0x1＋(B,L,N)	B個前からL個の文字データ列を N回繰り返す
RP(B, N) ポインタ制御命令	「11」＋0x2＋(B,L)	B個前ポインタをN回繰り返す
RDP(B, L, N) 出力データ列制御命令	「11」＋0x3＋(B,L,N)	B個前からL個の出力データ列を N回繰り返す
OMD(B, L, M, C) データ制御命令	「11」＋0x4＋(B,L,M,C)	B個前L個の文字データ列のM番目の データをCに変更し出力する
CP(B1, B2) ポインタ制御命令	「11」＋0x5＋(B1,B2)	ポインタP1とP2が指す文字データ列を 結合して出力する
⋮	⋮	⋮
SPW パスワード設定命令	「11」＋0xD＋(PW)	パスワードPWを設定する
SCR 著作権情報設定命令	「11」＋0xE＋(CR)	著作権情報CRを設定する
SCMコメント	「11」＋0xF＋(CM)	コメントCMを挿入する

但し、B:7ビット　N:4ビット　L:8ビット　M:8ビット　C:8ビット　とする。

具体例2の命令の作成例の説明図

出証特２００３－３０４２８０７

特２００３－０４５６０１

命令の種類	コード	意義
SD(B, L, M) 文字列定義命令	「11」 + 0x8 + (B,L,M)	B個前からL個の文字列を M番と指定する
SDO(B, L, M) 文字列定義出力命令	「11」 + 0x9 + (B,L,M)	B個前からL個の文字データ列を M番と指定し、その文字列を出力する
PD(M) ポインタ定義命令	「11」 + 0xA + (M)	直前のポインタをM番と指定する
CS(M) コード置き換え命令	「11」 + 0xB + (M)	SD命令等で指定されたM番の文字列 コードに置き換える

但し、B:7ビット　L:8ビット　M:5ビット　とする。

具体例3の命令の作成例の説明図

【図１４】

(a)順位による打ち切り

順位(M)	文字列	ポインタ	出現頻度(T)
1	S1	P1	18
2	S2	P2	12
3	S3	P3	10
4	S4	P4	9
⋮	⋮	⋮	⋮
31	S31	P31	4
32	S32	P32	4
33	S33	P33	4

順位Mによる
打ち切り

(b)出現頻度による打ち切り

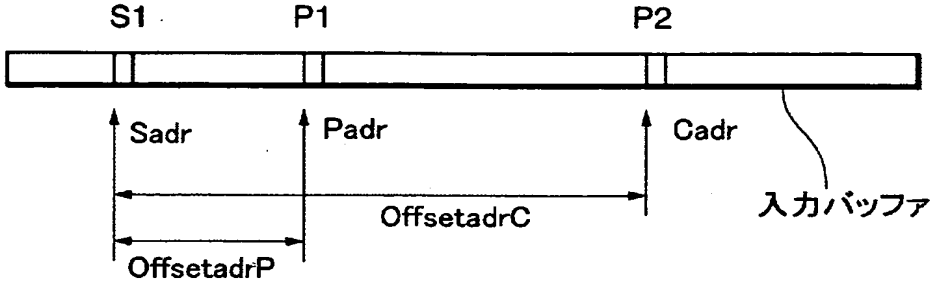
順位(M)	文字列	ポインタ	出現頻度(T)
1	S1	P1	12
2	S2	P2	8
3	S3	P3	7
4	S4	P4	6
⋮	⋮	⋮	⋮
16	S16	P16	3
17	S17	P17	2
18	S18	P18	2

頻度Tによる
打ち切り

同一の文字列の出現頻度とコード化との関係を示す説明図

【図１５】

判定条件1： S1の文字数同一
判定条件2： Cadr−Padr=OffsetadrC−OffsetadrP



S1:文字列
P1:S1を示すポインタ
P2:S1を示すポインタ
Sadr,Padr,Cadr:入力バッファ内の相対アドレス
OffsetadrP,OffsetadrC:オフセット値を示す相対アドレス

同一のポインタの判定の説明図

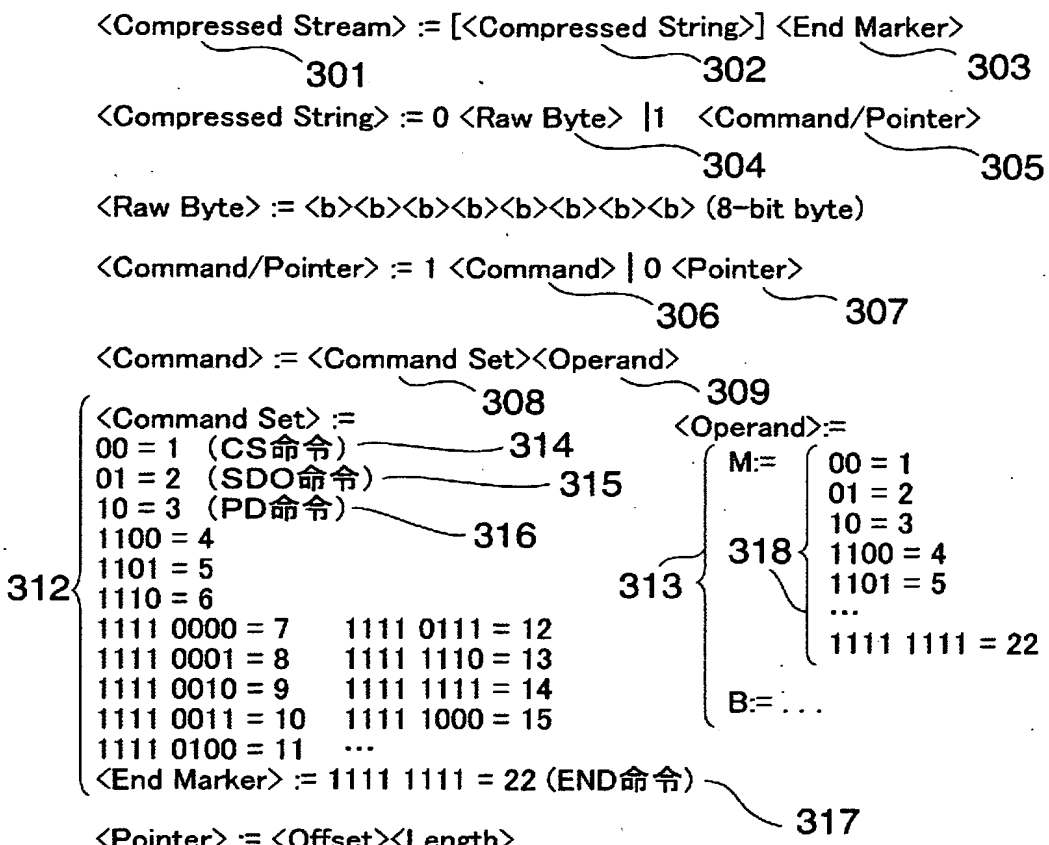
出証特２００３－３０４２８０７

【図１６】

指定順位(M)	定義命令	有効／無効フラグ
1	SDO(B,L,M)	1
2	SD(B,L,M)	1
3	PD(B,L,M)	1
4	SDO(B,L,M)	1
⋮		⋮
16	SDO(B,L,M)	1
17	——	0
⋮	⋮	⋮
32	——	0

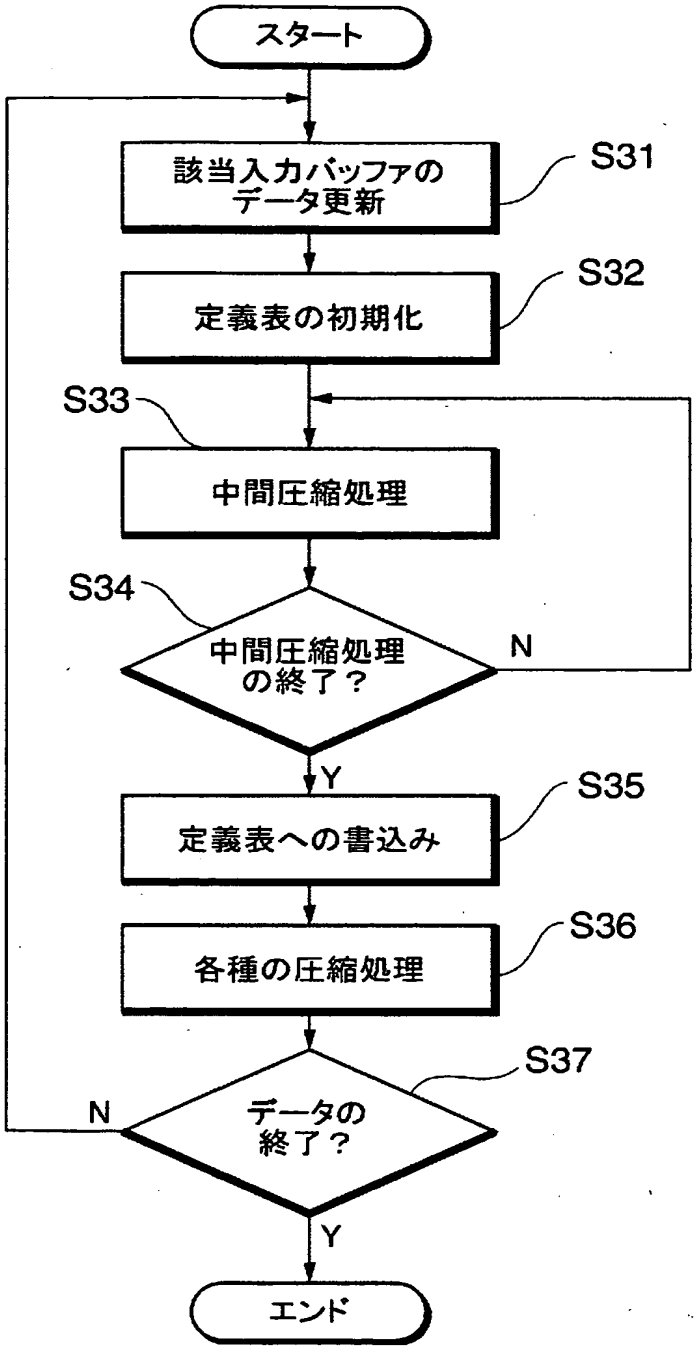
定義表の作成例の説明図

【図１７】



具体例3のコード化の説明図

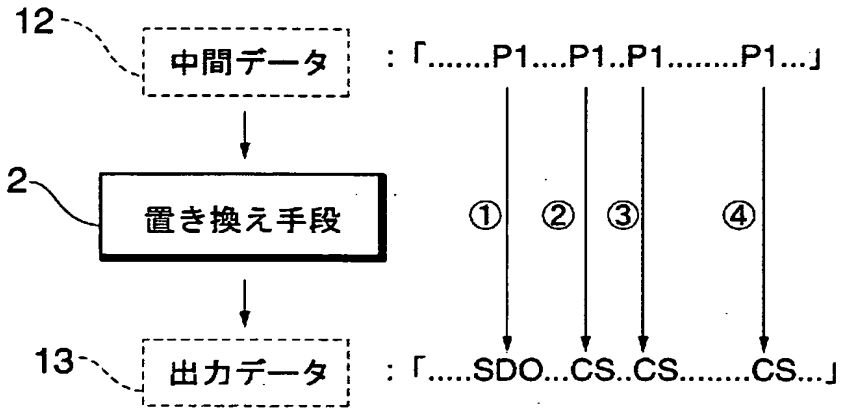
【図１８】



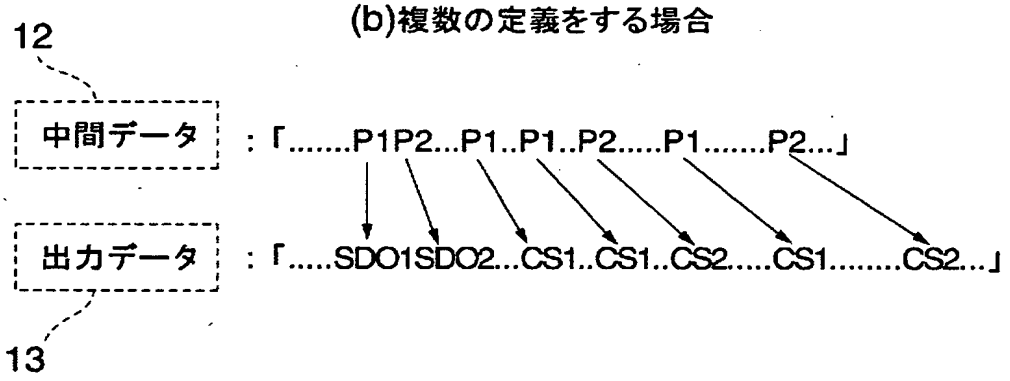
具体例3の動作フローチャート

【図１９】

(a)一の定義をする場合



(b)複数の定義をする場合



コード置き換え命令の説明図

【書類名】　　要約書

【要約】

【課題】　文字列以外の情報等を埋め込むといった自由な圧縮を行うことのできるデータ圧縮装置を実現する。

【解決手段】　圧縮装置１は、入力データ１１より中間データ１２として、文字データとポインタを出力する。置き換え手段２は、命令が対象とする文字列が、文字データの対象とする文字列とポインタが指示する文字列、または、これらの組合せと一致する場合に、文字データまたはポインタに置き換えて命令を出力し、文字データとポインタと命令を含む出力データ１３を出力する。

【選択図】　　図１

認定・付加情報

特許出願の番号	特願２００３－０４５６０１
受付番号	５０３００２９００５１
書類名	特許願
担当官	第七担当上席　　００９６
作成日	平成１５年　２月２５日

＜認定情報・付加情報＞

【提出日】	平成15年 2月24日

次頁無

出　願　人　履　歴　情　報

識別番号　　〔０００００２９５〕

１．変更年月日　　１９９０年　８月２２日
［変更理由］　新規登録
住　所　　東京都港区虎ノ門１丁目７番１２号
氏　名　　沖電気工業株式会社